Using SCXML to integrate Semantic Sensor Information into Context-Aware User Interfaces

Álvaro Sigüenza[1], José Luis Blanco[1], Jesús Bernat[2] and Luis A. Hernández[1]

[1] ETSI Telecomunicación - Universidad Politécnica de Madrid
28040 Madrid, Spain
{alvaro.siguenza,jlblanco,luis}@gaps.ssr.upm.es

[2] Telefónica R&D
28043 Madrid, Spain
bernat@tid.es

Abstract. This paper describes a novel architecture to introduce automatic annotation and processing of semantic sensor data within context-aware applications. Based on the well-known state-charts technologies, and represented using W3C SCXML language combined with Semantic Web technologies, our architecture is able to provide enriched higher-level semantic representations of user's context. This capability to detect and model relevant user situations allows a seamless modeling of the actual interaction situation, which can be integrated during the design of multimodal user interfaces (also based on SCXML) for them to be adequately adapted. Therefore, the final result of this contribution can be described as a flexible context-aware SCXML-based architecture, suitable for both designing a wide range of multimodal context-aware user interfaces, and implementing the automatic enrichment of sensor data, making it available to the entire Semantic Sensor Web.

1 Introduction

The rapid development of sensors and sensor networks makes it possible to capture an increasing number of physical contexts for modeling real world situations and enabling localized interaction [1]. As "*context is any information that can be used to characterize the situation of an entity"* [2],a countless number of context-aware applications are being proposed to assist users in everyday life activities. The environment itself becomes an essential part of the user-interface [3] and new ways of adaptation to the context are possible [4]: content presentation, content delivery and service adaptation.

But information gathered by sensors can be of distinct nature and is generally developed as particular solutions to customized applications. Thus, application-specific procedures are also required to design and develop context-aware user interfaces, making it difficult to extend or reuse them for evolving toward different application scenarios and services. Three main issues should be addressed:

1. Heterogeneous sensor data should be represented in a homogeneous way to make it available to different application domains.

2. Sensor data describing different contexts should be processed to identify specific contexts and situations relevant to the user, and from which applications could take advantage of.
3. Situations identified as relevant should be available to the *Interaction Manager*, i.e. the agent in charge of controlling the behavior of the user interface.

In this work, we present a framework to include information from heterogeneous sensor data into user interfaces based on two mayor components: 1) Semantic Web technologies: to enrich raw sensor data with semantic information; and 2) State-Chart machines: as a general process control mechanism for the orchestration and combination of different sources of information.

Semantic Web technologies, as an evolving extension of the current web, are intended to provide well-defined and machine accessible information, shareable across applications. Based on Ontologies, or formal descriptions of concepts and their relationships, applications may benefit from a shared semantic representation of sensor data [5]. At present, most widely used ontology languages are W3C recommendations: Resource Description Framework (RDF), Resource Description Framework Schema (RDFS), and Web Ontology Language (OWL), all based on XML.

The use of Semantic Web in user interfaces is not new. It has been mainly proposed for representing user profiles and preferences [6], or to provide a general model of the physical environment [7], but only little attention has been paid to sensor-based user interface adaptation [8]. Furthermore, just by defining ontologies, what is itself a complex task, is not enough to take full advantage of sensor data. Mechanisms are needed to process low-level data, generate higher levels of context information and finally identify relevant user's situations. To this end, two complementary tasks need to be addressed [9]: 1) Semantic Sensor data annotation: embedding semantic information into sensor data; 2) Semantic sensor data processing: process sensor data so that high-level information can be generated and shared among different applications.

Several approaches have been already proposed to implement these two tasks in an automatic way. In Noguchi et al. [10] raw sensor data is described in RDF to implement behavior detection services for validating automatic generation and connection of program components for sensor data processing in network middleware. The Semantic Web Architecture for Sensor Networks (SWASN) presented in [9] proposes the use of rules, based on local ontologies, to annotate domain-specific sensor data while a global sensor ontology is used for querying and inference. Also relevant are some international projects working on open architectures for smart environments as FP7 SENSEI [1] or Artemis SOFIA [11] EU projects. Differently from these approaches, as we are interested in the integration of semantic sensor data into user interfaces, our approach is based on the use of the general process control mechanism provided by state-chart machines. So far, in our approach, state-chart machines will be the framework to both semantic sensor data annotation and processing.

The rationale for this approach is that State-Charts are being successfully used to design and implement *Interaction Management* for the integration of different interaction modalities (keyboard, mouse, speech, vision, ink, haptics, etc.) in multimodal Human-Machine Interfaces (HMIs) [12]. What we propose here is to extend these capabilities of state-charts for representing, processing and integrating several input and output modalities, towards the annotation and processing of semantic sensor data.

In particular, we will explore the possibilities of the W3C standard, State Chart eXtensible Markup Language, SCXML [13], which provides a generic event-based state-machine execution environment based on Harel statecharts [14]. SCXML is intended to keep application's interaction management flow and both presentation and interpretation processes independent one from the others. Accordingly, when designing multimodal user interfaces, application developers can have a great flexibility by orchestrating several state-charts to process and combine different modalities of user inputs and outputs. In our architecture we will also use SCXML to semantically annotate sensor data, process it and provide specific user's contexts or situations to the *Interaction Manager*. The final result will be a flexible SCXML-based architecture suitable for both the design of a wide range of multimodal context-aware HMI interfaces, and the automatic enrichment of sensor data making it available to the Semantic Sensor Web.

The rest of the paper is organized as follows: Section 2 presents SCXML as a technology to implement multimodal HMI applications. In Section 3, we propose a SCXML-based high-level architecture to semantically annotate and process sensor data. Section 4 analyses a use case to illustrate how the proposed Semantic Sensor Web architecture can be used to implement a context-aware user interface for an in-car driving scenario. Finally, conclusions and future work are discussed in Section 5.

2 SCXML for Multimodal HMI

A simplified view of a multimodal HMI is depicted in Figure 1. Several input modalities from the user (mouse, keyboard, ink, speech recognition …) are processed and combined by an Input Handler, while the Presentation component combines several output modalities (graphics, images, speech synthesis…) to the user. Based on both input and output semantic representations the Interaction Manager will control the interaction flow with the user.

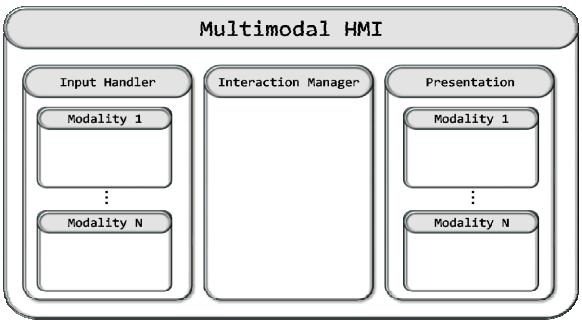

Fig. 1. Simplified view of a multimodal HMI depicted as a combination of state-chart machines

Conventionally, interaction management for user interfaces are conceived as set of possible states, in which different information is presented to the user, while the transition onto the next state is based on the actual user's input information. There are different approaches to the development of interaction managers, which can be broadly classified into: deterministic, based on rules for interpreting each user input and

updating the interaction state, or stochastic, where state transition probabilities are estimated using machine learning techniques [15].

In this work we will follow the deterministic approach, so, as in [12], based on SCXML the multimodal interaction is decomposed into a finite sequence of states (and consequently in a finite number of transitions), so a complete interaction will be the result of concatenating consecutive interaction turns in a proper way. Using SCXML, each state represents a different interaction situation that is reached through the recognition of events coming from the Input Handler. To give an example, the XML below describes a use case in which part of an in-car phone call application is implemented. By now we will not pay attention to the driver, car or driving contexts.

```
<?xml version="1.0"?>
 <scxml xmlns="http://wwww.w3.org/2005/07/scxml" version="1.0" initial-
      state="init">
   <state id="init">
     <transition event="machine.initiated" target="selectAction"/>
   </state>
   <state id="selectAction">
     <transition event="action.selectContact" target="selectContact"/>     …
     <transition event="action.selectNumber" target="selectNumber"/>
     <transition event="action.addContact"        target="addContact"/>
   </state>
   <state id="selectNumber">
     <transition event="action.call" target="phoneCall"/>
     <transition event="action.sendSMS" target="sendSMS"/>
   </state>
   <state id="selectContact"/>
   <state id="addContact"/>
   <state id="phoneCall"/>
   <state id="sendSMS"/>
</scxml>
```

In this example the application starts when the user decides to check his/her contact list. This action is mapped to a "*machine.initiated"* internal event within the state machine, which causes a transition onto "*selectAction*" state. By entering this state, the application displays a list of available actions the user may perform. The user can then select a specific action, causing the input handler to generate the corresponding event for this action. So, it fires the "*action.selectContact*" event to select a contact from the list, the "*action.selectNumber*" to dial a phone number, or rather the "*action.addContact*" event to add a new contact to the directory. Finally, suppose the user decides to choose a phone number, then the flow would enter the "*selectNumber*" state, so the user can do a phone call or send a SMS to this particular number.

Besides of using SCXML to control the interaction, several functionalities beyond general finite-state machines makes SCXML suitable to integrate different sources of information in a rather straight-forward way. SCXML allows defining state charts in a hierarchical way, so a state-chart can itself be a state within another state-chart (i.e. states composition). Also several state charts can run in parallel (i.e. simultaneous states) so it can process different sources of information at the same time. As Figure 2 illustrates, this hierarchy and concurrency can be used to program the Input Handler of a HMI for processing and combining different input modalities. In that way, when the Input Handler obtains a high-level semantic representation of the input, it is fired as an event towards the interaction-manager state-chart, shall force it to role.

Once the machinery has been set into motion, a number of evaluations and internal transitions might follow while deciding the next action to be taken. But in the end, a

stable state is to be entered in which the system shall remain until a new input arrives. By that time, the following action is sent to the Presentation module while the Interaction Management system waits. The multimodal Presentation module, also in SCXML, decides which modality (or combination of modalities) must be used. It is worth noting that, although not specifically considered in this work, state-charts can also be used for controlling actuators or actuation processes. In fact, the new VoiceXML version [16] is said to be implemented on top of this SCXML framework; which is just another sample to prove SCXML versatility.

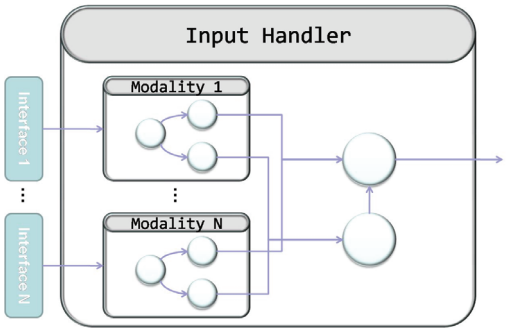

Fig. 2. Input Handler of a Multimodal HMI where state-chart machines are hierarchically combined to produce a high-level semantic representation of the user's multimodal input.

3 Semantic Sensor Web using SCXML

Current Sensor Web initiatives, such as OGC SWE Semantic Web Enablement [17], provide ubiquitous access to huge amounts of sensor data on the Web. Sensor Web usually lacks of semantic-level representation making it difficult to interoperate across different applications' domains. In order to address this problem, Sensor Web and Semantic Web technologies are being combined into what is called Semantic Sensor Web, as is proposed in recent initiatives such as the W3C Semantic Sensor Network Incubator Group [18]. The resulting capabilities of the Semantic Sensor Web for enriching sensor data with semantic annotations offer a great potential for designing real world context-aware applications and services.

In this work we propose the use of SCXML technology as a mechanism to enrich and process low-level heterogeneous (Sensor Web) data producing higher semantic representations of context information (Semantic Sensor Web). In that way semantic sensor data and context information will also be easily integrated into multimodal HMI interfaces described in the previous Section.

3.1 High-level architecture

Figure 3 represents the high-level description of our architecture ranging from the Sensor Web and the Semantic Sensor Web, and toward a context-aware HMI designed, by using SCXML.

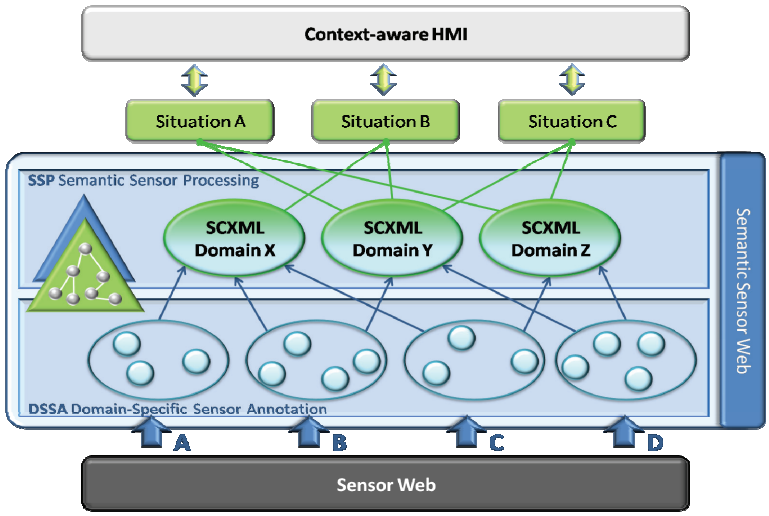

Fig. 3. High-level architecture for Semantic Sensor annotation and processing using SCXML

Beginning from our lower layer, Domain-Specific Sensor Annotation layer (DSSA), heterogeneous sources of sensor data, from Sensor Web infrastructures, are processed by domain-specific SCXML machines. These are designed relying on local ontologies so they can semantically annotate events related to specific sensor data. In a second layer, Semantic Sensor Processing layer (SSP), the hierarchy and concurrency functionalities supplied by SCXML are used to process information from different local domains generating higher levels of semantic information. As in the lower DSSA layer, in SSP layer state-charts machines also need to be designed on top of specific ontologies, although in this level they must represent more global or wide domains. Three main outcomes can be provided by the SSP layer:

1. *Semantically annotated sensor data*: as the SSP layer processes sensor data already annotated in the DSSA layer, one possible outcome in the SSP layer can be just to extend the already semantically embedded information in DSSA with higher-level semantics from SSP.
2. *Semantically annotated virtual sensor data*: as SSP can integrate or process semantic sensor data from different local domains in DSSA level, it can produce new real-world related information that can be considered as virtual sensor data.
3. *Detection of specific situations*: as a result of processing information from the DSSA layer, the particular states of the SCXML machines at the SSP layer can be interpreted as specific higher-level situations derived from local domains. Thus, certain patterns can be identified within the actual situation, which are meaningful in themselves.

As shown in the Figure, semantic annotation at DSSA and SSP layers (outcomes 1 & 2) can be considered a mechanism for exposing sensor data from Sensor Web to the Semantic Sensor Web. While outcome 3, context and situation detection, represents a valuable source of high-level information easily integrable into HMI interfaces based on SCXML. This annotated semantic sensor data could also be useful for interfaces or applications implementing strategies based on semantic queries or inference engines.

3.2 Semantic Sensor data annotation and processing

SCXML is a general process control language [13], so it is suitable for processing, however it is still to be stated how semantic annotated data can be managed. SCXML provides a standard way to define a data model to represent and store common data used by several state machines. This data model can be used to evaluate guard conditions when a transition in the machine is required. The data model consists of a `<datamodel>` label containing several elements which define variables, each one containing specific information expressed in a XML format. In our architecture, as it will be detailed in Section 4, raw sensor data represented in XML is initially stored in the SCXML data model, and, as a result of state-chart processing, semantic annotations are embedded in the original data and stored again in the data model to enrich it. This process is implemented at both DSSA and SSP layers so that the SCXML data model may include different levels of semantic representations.

Another important implementation issue to consider is how semantic sensor data annotation and processing are managed as events drive most SCXML transitions. When a sensor experiences a change in its context, it sends events to communicate it to the architecture. In its first DSSA layer, these events drive a set of state machines, able to model the local context they are related to. Once a state machine reaches a certain state, it invokes a process which is in charge of annotating this data with semantic information referencing an ontology about the domain represented by the state machine, and storing the data in the SCXML datamodel. Thus, a semantic link between data and a universal representation of this data is established, allowing applications and other state machines to automatically understand data from a great variety of heterogeneous sources, and enable them to process and infer new semantic data.

Changes occurred in the lowest domains are communicated to the next level by sending an event indicating its datamodel has been updated. The following level, the SSP layer, is able to take advantage of SCXML guard conditions to specify inference rules that allow inferring higher semantic data. Using these rules it is possible to integrate low-level information domains into high-level domains, defining new "virtual sensors" that were not available in the physical world.

Finally, taking advantage of the SCXML parallel and hierarchical processing capabilities, state machines are continually merging data and building higher-semantic context domains. Those are recursively merged until we are capable of describing full context situations which could be useful to the context-aware interactive applications we are to design.

4 Use Case analysis

In this Section, a context-aware user interface for an in-car driving scenario will be used as a motivating example to illustrate the specific SCXML mechanisms we have implemented. A series of intrinsic peculiarities have been identified within the in-vehicle scenario, which appears to be quite suitable for the proposed architecture:

- A great variety of heterogeneous data sources are available in the driving environment (vehicle, road, traffic, weather…). There are multiple in-vehicle sensors which represent different context variables, but also other external

sensor sources are available, provided by the *Sensor Web*. The integration of both types of sources constitutes a true challenge for present architectures.

- New infotainment and interactive in-vehicle multimodal systems have recently appeared. Managing these systems requires adapting their behavior according to the actual driver's context.
- Driving is a suitable environment to integrate contextual information (both local and external) with interaction information.

First the interface adaptation is described (subsection 4.1) and then the composition of sensor data at different semantic levels is discussed (4.2).

4.1 User Interface Adaptation

Following the example introduced in Section 2, if contextual information about driving situation is available, a context-aware user interface could be adapted. According to this, new context information will reach the *Interaction Manager*, besides the user's input information. Therefore the basic HMI scheme of Figure 1 is extended to the one depicted in Figure 4.

Thereby, while the *Interaction Manager* is working, an event from the *Input Handler* arrives, indicating the user wants to carry out a *selectNumber* action. Meanwhile, a "*dangerous driving situation"* is likely to appear in the Context Situation state machine, which will cause the corresponding event to be fired. If such situation appears, following with the Nishimoto's studies addressed in [19], the *Interaction Manager* should halt the interaction, at least until this dangerous situation disappears.

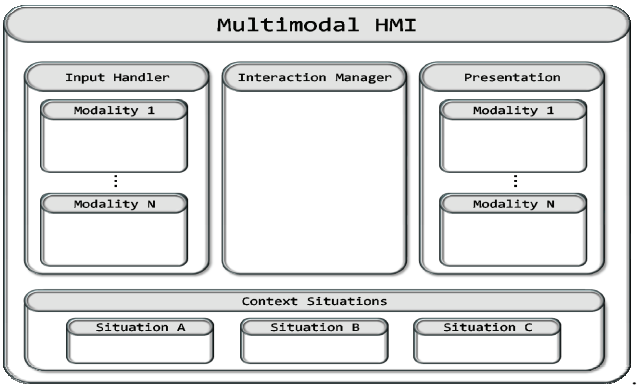

Fig. 4. General view of a context-aware multimodal HMI system, based on the combination of multiple state-chart machines.

Thus, the state machine must include a set of new states describing the expected curse for the interaction flow once such situations arise. We now include a snippet about the new interaction specification, as an extension to the one shown in Section 2.

In this example, when a "*dangerousDriving.situation*" event arises, the *Interaction Manager* shall move into the *interruptInteraction* state to indicate the *Presentation Manager* that no content should be presented to the user. Once the dangerous situa-

tion disappears (as the "*normal.situation*" event arrives) the state machine shall go back to the previous state (the *history* element is a SCXML element which allows saving the identifier for last state before the last transition).

```
<scxml     xmlns="http://wwww.w3.org/2005/07/scxml"     version="1.0"     initial-
  state="init">
   <state id="init">
     <transition event="machine.initiated" target="selectAction"/>
   </state>
   <state id="selectAction">
     <transition event="action.selectContact" target="selectContact"/>
     <transition event="action.selectNumber" target="selectNumber"/>
     <transition event="action.addContact"       target="addContact"/>
 <transition event="dangerousDriving.situation" target="interruptInteraction"/>
   </state>
   <state id="selectNumber"/>
   <state id="selectContact"/>
   <state id="addContact"/>
   <state id="interrumptInteraction">
     <transition event="normal.situation" target="history"/>
   </state>
</scxml>
```

4.2 Situation Description

In the driving context multiple sensors are available. The literature identifies three independent domains describing the global context for the driver-vehicle interaction: driver, vehicle and environment [20]. In our example, we focus on the traffic and weather sensors (provided by external *Sensor Web* resources) and the road type and vehicle sensors (virtual and physical local sensors).

To annotate the collected sensor data we have followed the Sensor Web Enablement initiative [17]. SWE is a group formed by the Open Geospatial Consortium in order to specify interfaces and meta-data encodings with the aim of integrating heterogeneous sensor data so that they could be accessible and controllable via Web. We have selected the Observations & Measurements language (O&M) [21] which defines a XML schema for describing and encoding real-time observations and measurements of sensor data. According to it, observation is an *act of observing a property or phenomenon, with the goal of producing an estimate of the value of a feature of interest* [21]. The standard also defines a large number of terms needed to describe measurements and the relationships among them. The following snippet represents an annotated measure from a wind speed sensor.

```
<swe:DataRecord>
    <swe:field name="WindSpeed">
        <swe:Quantity definition="urn:ogc:def:property:OGC:WindSpeed">
        <swe:uom code="mph"/>
        <swe:value> 90.0 </swe:value>
        </swe:Quantity>
    </swe:field>
</swe:DataRecord>
```

When there is a change on a certain feature of interest, a sensor fires an event towards the framework, encoded using O&M. This data updates the state machines

representing the three previously presented independent domains forming the DSSA level: driver, vehicle and environment. Each domain contains other domains (and therefore other state machines) which are paired with the sensors defined in the context. Thus, the vehicle domain is formed, for instance, by a *velocity* state machine and a *steering wheel angle* state machine, while the environment domain is defined by a *traffic road* state machine and a *weather* state machine (see SCXML example below). Each machine was previously registered to the events coming from certain sensors related to the local context, so they only receive data associated to subscribed events.

```
<parallel id="context">
  <parallel id="driver/>
  <parallel id="vehicle">
    <state id="velocity"/>
    <state id="steeringWheelAngle"/>
  </parallel>
  <parallel id="environment">
    <state id="trafficRoad"/>
    <parallel id="weatherConditions">
       <state id="speedWind"/>
    </parallel>
    <state id="roadType"/>
  </parallel>
</parallel>
```

Taking the *speed wind state* machine as example, we could specify it by introducing the following SCXML code.

```
<state id="speedWind">
  <initial/>
  <state id="StrongWind">
     <invoke targettype="semanticAnnotation" src="StrongWind"/>
  </state>
  <state id="ModerateWind">
     <invoke targettype="semanticAnnotation" src="ModerateWind"/>
  </state>
  <state id="LightWind">
     <invoke targettype="semanticAnnotation" src="LightWind"/>
  </state>
  <transition event="wind.change" cond="speedWind lt 30" target="LightWind"/>
  <transition event="wind.change" cond="speedWind gt 30 and lt 60" tar-
    get="ModerateWind"/>
  <transition event="wind.change" cond="speedWind gt 60" target="StrongWind"/>
</state>
```

Here, the state machine examines the value of the speed wind measure when a new event arrives. According to its value a transition to the corresponding target (*StrongWind, ModerateWind or LightWind*) might follow. So it enters a new state, where an external process is invoked to annotate the received O&M sensor data using a domain-specific ontology associated to that particular state machine.

We use RDFa (or Resource Description Framework –in- attributes) to embed semantic annotations into XML data. RDFa [22] is a W3C Recommendation that allows embedding rich metadata within Web documents. Thus, RDF attributes can be easily added to *O&M* data to provide the desired semantic annotations. We use the RDF *about* attribute to establish which resource is referenced by means of an ontology. In the next example, we demonstrate how the *windSpeed* data is said to be related to a weather ontology associated to the weather state machine domain.

```
<swe:DataRecord>
  <swe:field name="WindSpeed">
     <swe:Quantity definition="urn:ogc:def:property:OGC:WindSpeed"
       rdf:about="http://www.csd.abdn.ac.uk/research/AgentCities/WeatherAgent/we
       ather-ont.daml#windSpeed">
     <swe:uom code="mph"
       rdf:about="http://www.csd.abdn.ac.uk/research/AgentCities/WeatherAgent/we
       ather-ont.daml#mphSpeed">
     <swe:value> 90.0 </swe:value>
        </swe:Quantity>
   </swe:field>
</swe:DataRecord>
```

When the annotation process is completed, the state machine fires an event indicating the nature of the wind speed (*strong*, *moderate* or *light*), and stores the enriched data labels within the SCXML datamodel so that applications may consult them, and use them to their own benefit. The same process will be carried out when the lowest domains (DSSA layer) detect a high vehicle velocity or a dense traffic condition.

Above the DSSA, the SSP layer is in charge of providing higher or more abstract information by integrating data from different domains and/or by detecting specific situations that could be useful for the context-aware interface. The following SCXML fragment is an example of the implementation of SSP layer.

```
<state id="normalDriving">
  <transition cond="Data(environment, weather/speedWind) eq 'strong' and
      Data(environment, roadType) eq 'lane' and Data(vehicle, velocity) eq 'exce-
      sive'" target="dangerousDriving" />
</state>
<state id="dangerousDriving">
  <onentry>
     <send event="dangerousDriving.situation"/>
  </onentry>
</state>
```

Here, the SSP level is composed by two states representing two different and exclusive situations: *normalDriving* and *dangerousDriving*. The state machine will remain in the *normalDriving* state until the transition condition is fulfilled. Just before that, the DSSA layer will update the datamodel fields about *speedWind*, *roadType* and *velocity* and send an event to the SSP layer signalling a change in one of its variables. When the condition is met, a transition to the *dangerousDriving* state occurs and causes an event to be fired in order to inform the context-aware applications. Finally, this new situation is annotated in the datamodel according to an ontology, and the whole framework waits for the situation to change.

5 Conclusions and Future Work

Due to the development of new Sensor Web initiatives, access to our physical contexts becomes easier and the number of context-aware applications increases. However, sensors and sensor networks are undoubtedly heterogeneous, just as the information to be sensed. On the other hand, human-machine interfaces have been traditionally developed based on the user and the application, but have frequently missed the dramatic influence of context while interacting. New paradigms in programming and

applications' design have shed some light at this point, and suggested to bring both context and application together into the so called context-aware applications.

In this paper we have presented a novel architecture, based on the SCXML framework, which focuses on this particular issue by considering the semantic annotation and processing of sensed data in order to provide higher-level semantic information, which can be easily used at the higher HMI level. By reviewing the conventional finite-state interaction flow model to be implemented according to the SCXML standard, we have pointed out how easy it is to design an application. Introducing SCXML designs to handle both inputs and outputs seems also a simple process, and guarantees full compatibility across HMI components in the framework. SCXML formal conception also includes a set of relevant properties and structures, from which we take advantage for the annotation and processing of sensor data. At this level, the proposed scheme relays on the ontological description of specific domains to annotate data (DSSA layer). The subsequent annotation process becomes knowledge intensive as semantic relations weave a complex network by combining information from various domains (SSP layer). The final result is locally stored to outline the actual context image, and so that context-aware applications can freely access the whole set of annotations. Furthermore, particularly relevant situations can be identified within the actual situation, and pushed up to the application layer. The proposed architecture has been implemented on top of the OSGi framework, and proved to be extremely flexible and robust. The use case described in Section 4 is in fact a real scenario we have considered, though further tests must be performed to validate the interaction strategies introduced.

Future work shall discuss the limitations imposed by state-chart schemes, i.e. the difficulty in both managing and designing, particularly as no graphical development environment is yet available for SCXML. Combining multiple sources of information is also a tough task, especially when the number of sources and the amount of information is potentially very big. Scalability is therefore extremely important for this kind of infrastructures and applications, as well as granting access to the Semantic Sensor Web both to retrieve and share information. This is especially intriguing for in-vehicle frameworks, as quality of service is not always granted.

6 Acknowledgements

The activities described in this paper were funded by the Spanish Ministry of Science and Technology as part of the TEC2009-14719-C02-02 project.

References

1. Presser, M., Gluhak, A., Krco, S., Hoeller, J. and Herault, l.: SENSEI - Integrating the Physical with the Digital World of the Network of the Future, 17th ICT Mobile Summit, Stockholm, Sweden, 10-12 June (2008)
2. Dey,A.K., Abowd, G.D., and Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16, 97-166 (2001)

3. FIDIS deliverable, D7.3: "Report on Actual and Possible Profiling Techniques in the Field of Ambient Intelligence", Editors: Wim Schreurs, Mireille Hildebrandt (VUB, Belgium), Mark Gasson, Kevin Warwick (Reading University, UK), August (2005).
4. Sathish, S.: Using Declarative Models in Multi-device Smart Space Environments, Workshop on Declarative Models of Distributed Web Applications, Dublin, Ireland, http://www.w3.org/2007/02/dmdwa-ws/talks/sathish.pdf June (2007)
5. Berners-Lee, T., Lendler, J. and Lassila, O.: The Semantic Web, Scientific American, vol. 284, no. 5 (2001)
6. Angioni, M., Demontis, R., Deriu, M., De Vita, E., Lai, C., Marcialis, I., Paddeu G., Pintus, A., Piras, A., Sanna, R., Soro, A., and Tuveri, F.: A Collaborative, Semantic and Context-Aware Search Engine. In -, editor, Proc. Of ICEIS 2007 - Software Agents And Internet Computing. Volume 1, (2007)
7. Mohamed A. F., Mounir M., Ismail K. I.: A Novel Approach for Ontology Distribution in Ubiquitous Environments, International Journal of Web Information Systems, Vol. 2 Iss: 3/4, pp.225 – 231 (2007)
8. Bell, D., Heravi, B. and Lycett, M.: Sensory semantic user interfaces (SenSUI), 2nd International Workshop on Semantic Sensor Networks 2009, Washington, October (2009)
9. Huang, V. and Stefanov, S.: Semantic Sensor Information Description and Processing, Proceedings of the Second International Conference on Sensor Technologies and Applications, pp. 456-461 (2008)
10. Noguchi, H., Mori, T., and Sato, T.: Automatic Generation and Connection of Program Components based on RDF Sensor Description in Network Middleware, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2008-2014 (2006)
11. SOFIA project – Smart Objects For Intelligent Applications, funded through the European Artemis program, http://www.sofia-project.eu/
12. Wilcock, G. and Jokinen, K.: SCXML, Multimodal Dialogue Systems and MMI Architecture, Workshop on W3C Multimodal Architecture and Interfaces, Fujisawa, Japan (2007)
13. W3C: State Chart XML (SCXML): State Machine Notation for Control Abstraction, http://www.w3.org/TR/scxml/, W3C Working Draft 13 May (2010)
14. Harel, D. (1987). StateCharts: A visual Formalism for Complex Systems. In: Science of Computer Programming 8, North-Holland.
15. Young, S.: Cognitive User Interfaces" IEEE Signal Processing Magazine,27(3): 128-140, (2010)
16. W3C: Voice Extensible Markup Language (VoiceXML) Version 2.0, W3C Recommendation 16 http://www.w3.org/TR/voicexml20/ March (2004)
17. OGC® Open Geospatial Consortium, Inc. : Sensor Web Enablement (SWE), http://www.opengeospatial.org/ogc/markets-technologies/swe
18. W3C Semantic Sensor Network Incubator Group, http://www.w3.org/2005/Incubator/ssn/
19. Nishimoto, T, Shioya, M., Takahashi, J., and Daigo, H.: A study on Dialogue Management Principles Corresponding to the Driver's Workload. Advances for In-Vehicle and Mobile Systems:251 -264. (2007)
20. Amditis, A., Kussmann, H., Polynchronopoulos, A., Engström, J., and Andreone, L.: System Architecture for integrated adaptive HMI solutions. In: Intelligent Vehicles Symposium. Tokyo, Japan (2006)
21. OGC® Open Geospatial Consortium, Inc.: Observations and Measurements (O&M), http://www.opengeospatial.org/standards/om
22. W3C: RDFa Primer. Bridging the Human and Data Webs. W3C Working Group Note 14 October 2008http://www.w3.org/TR/xhtml-rdfa-primer/